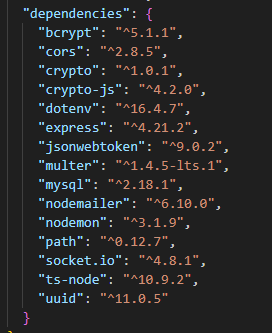
**Yourdle – Backend Fejlesztői Dokumentáció**

**Backend függőségei**

****

**Futtatás**

A program futtatásához az Api mappában telepítenünk kell a node modulokat. Ezt az „npm i” paranccsal tehetjük meg.

A szerver indítása előtt érdemes beimportálni a DB mappában található sql fájlt a phpmyadmin felületen.

Az Api-t lebuildelni az „npm run build” paranccsal tudjuk.

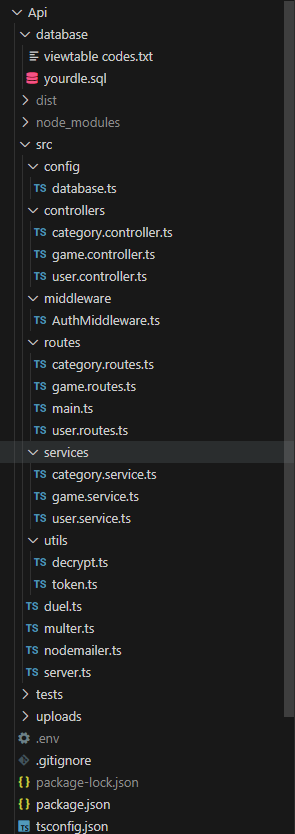
A szervert azt „npm start” paranccsal, a fejlesztői környezetet az „npm run dev”-el indítjuk el.

A backend a <http://localhost:3000> érhető el, miután futtatjuk.

**Futtatáshoz szükséges szoftverek:**

* node (20.19.0, vagy újabb)
* npm (10.2.3, vagy újabb)
* Visual Studio Code (1.67.2, vagy újabb)
* XAMPP (3.3.0, vagy újabb)
* Firefox (111.0.1, vagy újabb)
* Google Chrome (111.0.5.5, vagy újabb)

**1. Projekt Struktúra**

****

**2. Rendszer Áttekintés**

A Yourdle egy többjátékos típusú kérdezz-felelek játék, amely különféle feladványtípusokat kezel: klasszikus, emoji, idézet, leírás és kép alapú. A játék célja, hogy a játékosok minél kevesebb próbálkozással megfejtsék a feladványokat, miközben bővítik tudásukat és szórakoznak. A backend Node.js + TypeScript alapokra épül, Express keretrendszerrel.

**3. Szerverindítás (server.ts)**

A szerver konfigurálása:

* Express middleware-ek beállítása
* API útvonalak csatlakoztatása: app.use("/api", router)
* Képek statikus kiszolgálása: app.use('/uploads', ...)
* Multer alapú fájl feltöltés (autentikációval)
* Socket.IO integráció játékos párbajhoz
* Email kezelők: jelszó visszaállítás stb.

**4. Middleware – Auth**

JWT alapú hitelesítés:

const token = req.headers.authorization?.split(' ')[1];

const user = jwt.verify(token, process.env.JWT\_SECRET);

Hiányzó vagy hibás token esetén 400-as válasz.

**5. Játék API (game.routes.ts)**

| **Metódus** | **Útvonal** | **Leírás** |
| --- | --- | --- |
| GET | /allClassic/:id | Klasszikus kérdések |
| GET | /solutionClassic/:id | Klasszikus megoldás |
| GET | /allEmoji/:id | Emoji játék kérdések |
| GET | /solutionEmoji/:id | Emoji megoldás |
| GET | /allDescription/:id | Leírás alapú kérdések |
| GET | /solutionDescription/:id | Leírás alapú megoldás |
| GET | /allQuote/:id | Idézetek |
| GET | /solutionQuote/:id | Idézet megoldás |
| GET | /allPicture/:id | Képes feladványok |
| GET | /solutionPicture/:id | Kép megoldás |
| POST | /leaderboard | Ranglista frissítése (auth) |
| GET | /leaderboard | Teljes ranglista |
| GET | /leaderboard/:id | Egy felhasználó statisztikái |
| POST | /saveMatchResult | Játék eredmény mentése |

**6. Felhasználói API (user.routes.ts)**

| **Metódus** | **Útvonal** | **Leírás** |
| --- | --- | --- |
| POST | /login | Bejelentkezés |
| POST | /register | Regisztráció |
| PATCH | /profile | Profil frissítése (auth) |
| GET | /user/:id | Felhasználó lekérdezése (auth) |
| DELETE | /deleteProfilePicture/:id | Profilkép törlése (auth) |
| PATCH | /change-password/:id | Jelszó csere (auth) |
| GET | /history/:id | Meccstörténet (auth) |
| GET | /allUsers | Összes felhasználó (auth) |
| DELETE | /delete/:email | Felhasználó törlése (admin) |
| PATCH | /edit/:id | Felhasználó módosítása (admin) |

Felhasználói műveletek:

* Regisztráció: E-mail/jelszó validáció, hash-elt mentés
* Login: Jelszó ellenőrzés, JWT token generálás
* Profil szerkesztés: Email, név frissítése
* Jelszócsere: Régi ellenőrzés + új jelszó validálás
* Profilkép törlés: Fájlrendszerből és DB-ből törlés
* Meccstörténet: games\_vt nézetből lekérdezés

**7. Kategória API (category.routes.ts)**

| **Metódus** | **Útvonal** | **Leírás** |
| --- | --- | --- |
| GET | /allPublicCategories | Publikus kategóriák listázása |
| GET | /category/:id | Egy kategória adatainak lekérdezése |
| POST | /category | Új kategória létrehozása |
| POST | /classic | Klasszikus feladvány hozzáadása |
| POST | /description | Leírás feladvány hozzáadása |
| POST | /quote | Idézet feladvány hozzáadása |
| POST | /emoji | Emoji feladvány hozzáadása |
| POST | /picture | Kép feladvány hozzáadása |
| GET | /allCategory | Minden kategória lekérdezése (auth) |
| GET | /data/:id | Adott kategória összes feladványa |
| DELETE | /:id | Kategória törlése (auth) |

**8. Feladványok tárolása**

Minden feladványtípus egyedi adattáblába kerül mentésre:

* classic



* description:



* quote:



* emoji:



* picture:



Minden új adat uuidv4() ID-t kap.

**9. Kategória adatok lekérdezése**

A getCategoryDataService(id) függvény lekéri az adott kategóriához tartozó feladványokat:

1. Lekérdezi a kategória beállításait (milyen típusok engedélyezettek)
2. Lekérdezi a hozzá tartozó adatokat a megfelelő táblákból
3. A képeket visszafejti vagy placeholder-t ad

**10. Ranglista**

| **Művelet** | **Leírás** |
| --- | --- |
| uploadLeaderboardService | Eredmények mentése győzelem/vereség alapján |
| getAllLeaderboardService | Ranglista lekérdezése |
| getLeaderboardOneUser | Egy játékos statisztikája |

A képek visszafejtve jelennek meg vagy defaultként placeholder kerül beállításra.

**11. Adatbázis szerkezet (fő táblák)**

* users – 
* games – 
* leaderboard – 
* categories – 
* games\_vt – 

**12. Tokenek és hitelesítés**

* A bejelentkezés során JWT token generálódik (generateToken)
* Token tartalma: id, email, név, szerepkör, profilkép
* Token szükséges a legtöbb művelethez (authMiddleware)

**13. Validáció és biztonság**

* Jelszavak bcrypt-tel hash-eltek
* Regex alapú e-mail és jelszó ellenőrzés
* Token ellenőrzés minden védett útvonalon
* SQL injection elleni védelem: paraméterezett lekérdezések

**14. Fájlkezelés**

* multer kezeli a feltöltéseket (profilkép, kategória kép)
* Fájlok elérhetőek az uploads/ mappán keresztül
* Titkosított kép (pl. picture típus) visszafejtve kerül megjelenítésre



**15. Hibakezelés**

Minden endpoint ellenőrzi a bemeneti adatokat, pl.:

if (!email || !password) {

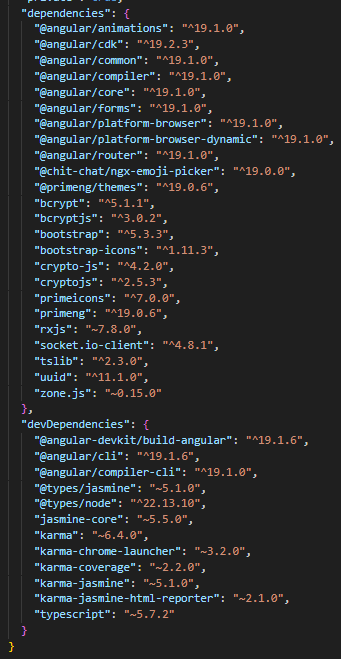
return res.status(400).json({ success: false, message: "Hiányzó adatok!"});

}

Ha a lekérdezés során hiba lép fel, a backend 500 Internal Server Error választ küld.

**Yourdle – Frontend Fejlesztői Dokumentáció**

**Frontend függőségei**

****

**Futtatás**

A program futtatásához a Yourdle mappában telepítenünk kell a node modulokat. Ezt az „npm i” paranccsal tehetjük meg.

Az applikációt lebuildelni az „npx ng build” paranccsal tudjuk.

Az applikációt azt „npx ng serve” paranccsal indítjuk, a fejlesztői környezetet az „npm start”-tal indítjuk el.

A frontend a <http://localhost:4200> érhető el, miután futtatjuk.

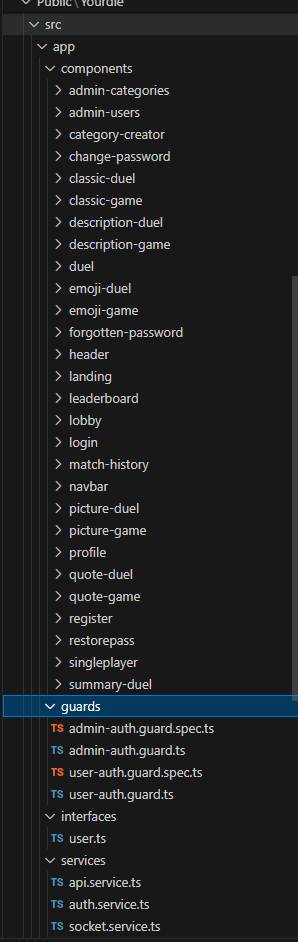
**Futtatáshoz szükséges szoftverek:**

* node (20.19.0, vagy újabb)
* Angular(19.1.6, vagy újabb)
* npm (10.2.3, vagy újabb)
* Visual Studio Code (1.67.2, vagy újabb)
* XAMPP (3.3.0, vagy újabb)
* Firefox (111.0.1, vagy újabb)
* Google Chrome (111.0.5.5, vagy újabb)

**1. Technológiai Stack**

* **Keretrendszer**: Angular (v14+ ajánlott)
* **UI komponensek**: PrimeNG
* **Layout**: PrimeFlex + saját CSS
* **Stílusvilág**: Cyberpunk témájú (sötét alapon neon színek)
* **Állapotkezelés**: Szervizek és BehaviorSubject, localStorage
* **Hitelesítés**: JWT token + AuthGuard
* **Kommunikáció**: HttpClient + REST API

**2. Projektstruktúra**



**3. Komponensek**

**Játék komponensek**

| **Komponens** | **Leírás** |
| --- | --- |
| classic-game | Klasszikus játék egyéni nézet |
| classic-duel | Klasszikus játék párbaj nézet |
| emoji-game, emoji-duel | Emoji alapú játékmenetek |
| description-game, description-duel | Leírás alapján zajló játék |
| quote-game, quote-duel | Idézet alapján zajló játék |
| picture-game, picture-duel | Képfelismerés játék |
| summary-duel | Párbaj összegzése, eredmény kijelzés |
| singleplayer | Egyjátékos mód |

**Auth és user komponensek**

* login, register, forgotten-password, restorepass
* profile, change-password

**Egyéb UI**

* landing, navbar, header, lobby, leaderboard, match-history

**Admin**

* admin-categories, admin-users, category-creator

**4. Routing és Guardok**

**guards/**

| **Guard fájl** | **Funkció** |
| --- | --- |
| user-auth.guard.ts | Bejelentkezett felhasználók védelme |
| admin-auth.guard.ts | Csak admin jogosultsággal rendelkező oldal |

Mindkét guard ellenőrzi a JWT token jelenlétét és szerepkört, pl.:

canActivate(): boolean {

const token = localStorage.getItem('token');

return !!token && this.userService.isAdmin();}

**5. Szervizek (services/)**

| **Szerviz fájl** | **Funkció** |
| --- | --- |
| api.service.ts | Backend API-hívások központi kezelése |
| auth.service.ts | Login, logout, token mentés |
| socket.service.ts | WebSocket kezelés (párbaj funkciókhoz) |

Az API-hívások az environment.apiUrl alapján működnek.

**6. Stílus és Dizájn**

**Cyberpunk Design:**

* **Sötét háttér** (#0d0d0d / #1a1a1a)
* **Neon akcentus színek**: #00ffff, #ff00ff, #39ff14
* PrimeNG témák átszínezve vagy override-olva SCSS-ben
* Komponens szintű stílusok (Angular component styleUrl)

**Responsiveness:**

* PrimeFlex + saját media query-k
* Minden nézet mobil-kompatibilis, flexbox és grid elrendezés

**7. Kommunikáció a Backenddel**

A frontend és a backend közötti adatcserét teljes egészében az ApiService osztály valósítja meg. Ez az Angular szolgáltatás központosítja az összes API-hívást, beleértve az autentikációt, a felhasználókezelést, a kategóriák és játékok kezelését is.

**Szerviz: api.service.ts**

Az ApiService a HttpClient segítségével hajtja végre a REST API-hívásokat. A tokenHeader() metódus automatikusan csatolja a Bearer token fejlécet a védett végpontokhoz.

tokenHeader(): { headers: HttpHeaders } {

const token = this.getToken();

const headers = new HttpHeaders({

'Authorization': `Bearer ${token}`

}); return { headers };}

**Token alapú autentikáció**

A JWT token a localStorage-ban tárolódik és automatikusan hozzáadásra kerül minden védett kéréshez.

**Backend hívások típusonként**

**Autentikáció**

* login(data)
* registration(data)
* forgottPassword(data)
* ChangePassword(id, data)

**Felhasználó kezelés**

* getUserById(id)
* profileSave(data)
* deleteProfilePicture(id)
* getAllUsers()
* editUser(id, data)
* deleteByEmail(email)
* uploadFile(profilePicture, id)
* MatchHistory(id)

**Kategóriák**

* getPublicCategories()
* getCategoryByID(id)
* createCategory(data)
* getAllCategory()
* getCategoryData(id)
* deleteCategory(id)
* createClassic(data)
* createDescription(data)
* createQuote(data)
* createPicture(data)
* createEmoji(data)
* uploadCategoryPicture(picture, answer, categoryID)

**Ranglista / játék eredmények**

* uploadLeaderboard(data)
* getLeaderboard()
* getLeaderboardOneUser(id)
* saveMatchResult(matchData)

**Játékadatok lekérése**

* getAllClassic(id)
* getSolutionClassic(id)
* getAllEmoji(id)
* getSolutionEmoji(id)
* getAllDescription(id)
* getSolutionDescription(id)
* getAllQuote(id)
* getSolutionQuote(id)
* getAllPicture(id)
* getSolutionPicture(id)

**8. Socket.IO integráció**

A socket.service.ts biztosítja a valós idejű kommunikációt a szerverrel (párbajhoz):

* Csatlakozás lobbyhoz
* Ellenfél találás
* Eredmények élő frissítése

**9. Állapotkezelés**

* Alapértelmezésben BehaviorSubject alapú állapotmegosztás (authService.user$, stb.)
* Token alapú bejelentkezés esetén localStorage-ben tárolás